Prototyping RISC-V Based Heterogeneous Systems-on-Chip with the ESP Open-Source Platform

Paolo Mantovani, Giuseppe Di Guglielmo, Davide Giri and Luca P. Carloni
Why ESP?

Heterogeneous systems are pervasive
Integrating accelerators into a SoC is hard
Doing so in a scalable way is very hard
Keeping the system simple to program while doing so is even harder

ESP makes it easy

ESP combines a scalable architecture with a flexible methodology
ESP enables several accelerator design flows and takes care of the hardware and software integration
ESP Vision: Domain Experts Can Design SoCs

Application Developers

- K
- PyTorch

Hardware Designers

- CHISEL
- SystemVerilog

HLS Design Flows

SoC Integration

RTL Design Flows

Rapid Prototyping

accelerator

RISC-V®

NVDLA.org®

accelerator

accelerator

* By Nvidia Corporation

** By lewing@isc.tamu.edu Larry Ewing and The GIMP
ESP Architecture

• RISC-V Processors
• Many-Accelerator
• Distributed Memory
• Multi-Plane NoC

The ESP architecture implements a distributed system, which is scalable, modular and heterogeneous, giving processors and accelerators similar weight in the SoC.
ESP Architecture: Processor Tile

• Processor off-the-shelf
  o RISC-V Ariane (64 bit)
  o SPARC V8 Leon3 (32 bit)
    o L1 private cache

• L2 private cache
  o Configurable size
  o MESI protocol

• IO/IRQ channel
  o Un-cached
  o Accelerator config. registers, interrupts, flush, UART, ...
ESP Architecture: Memory Tile

- **External Memory Channel**

- LLC and directory partition
  - Configurable size
  - Extended MESI protocol
  - Supports coherent-DMA for accelerators

- DMA channels

- IO/IRQ channel
ESP Architecture: Accelerator Tile

- Accelerator Socket w/ Platform Services
  - Direct-memory-access
  - Run-time selection of coherence model:
    - Fully coherent
    - LLC coherent
    - Non coherent
  - User-defined registers
  - Distributed interrupt
ESP Accelerator Socket

ESP Accelerator Socket

ESP accelerator
HLS [C/C++, SystemC, Tensorflow*, Pytorch*]
RTL [Chisel, Verilog, ...]

PLM

read/write config done

private cache TLB DMA ctrl cfg regs IRQ

coherence planes coherent-DMA planes DMA planes IO/IRQ plane

third-party accelerator
(NVDLA*, ...)

AXI4 bus APB bus IRQ

NoC

Third-Party Accelerator Socket*

Columbia University
In the City of New York
ESP Platform Services

### Accelerator tile
- DMA
- Reconfigurable coherence
- Point-to-point
- ESP or AXI interface
- DVFS controller

### Processor Tile
- Coherence
- I/O and un-cached memory
- Distributed interrupts
- DVFS controller

### Miscellaneous Tile
- Debug interface
- Performance counters access
- Coherent DMA
- Shared peripherals (UART, ETH, ...)

### Memory Tile
- Independent DDR Channel
- LLC Slice
- DMA Handler
ESP Software Socket

• ESP accelerator API
  o Generation of device driver and unit-test application
  o Seamless shared memory

```c
/*
 * Example of existing C application
 * with ESP accelerators that replace
 * software kernels 2, 3 and 5
 */
{
  int *buffer = esp_alloc(size);
  for (...) {
    kernel_1(buffer,...); /* existing software */
    esp_run(cfg_k2); /* run accelerator(s) */
    esp_run(cfg_k3);
    kernel_4(buffer,...); /* existing software */
    esp_run(cfg_k5);
  }
  validate(buffer); /* existing checks */
  esp_cleanup(); /* memory free */
}```
ESP Methodology In Practice

Accelerator Flow

- Generate accelerator
- Specialize accelerator
  * this step is automated
  * for ML applications
- Test behavior
- Generate RTL
- Test RTL
- Optimize RTL

SoC Flow

- Generate sockets
- Configure RISC-V SoC
- Compile bare-metal
- Simulate system
- Implement for FGPA
- Configure runtime
- Compile Linux
- Deploy prototype

* this step is automated
* for ML applications
ESP Accelerator Flow

Developers focus on the high-level specification, decoupled from memory access, system communication, hardware/software interface.
ESP Interactive SoC Flow

SoC Integration
ESP: A Flexible Platform for Open-Source Hardware

We hope that ESP will serve the OSH community as a Platform to develop software for RISC-V and accelerators for any application domain.
Thank you from the ESP team!

https://esp.cs.columbia.edu

https://github.com/sld-columbia/esp

System Level Design Group

Columbia University